Chapter 15

Getting Better Results With Latent Semantic Indexing

Preslav Nakov

Department of Mathematics and Informatics–Sofia University, and Rila Solutions, Bulgaria

preslav@rila.bg, preslav@rocketmail.com

Abstract. The paper presents an overview of some important factors influencing the quality of the results obtained when using Latent Semantic Indexing. The factors are separated in 5 major groups and analyzed both separately and as whole. A new class of extended Boolean operations such as OR, AND and NOT (AND-NOT) and their combinations is proposed and evaluated on a corpus of religious and sacred texts.

1 Introduction

The usage of Latent Semantic Analysis is a classical method for automatic indexing and information retrieval. Although it is a comparatively old and well-studied technique, there are several important problems that still remain unsolved. The main factors influencing the quality of results obtained by LSI can be separated in the following groups:

- Pre-processing and parsing
- Frequency matrix transformations
- Choice of dimensionality
- Choice of similarity measure
- Usage of Extended Boolean operations

2 Latent Semantic Indexing

The *Latent Semantic Indexing* (LSI) is a powerful statistical technique for information retrieval. The main idea of the method is that there is a latent link between the words and their context (phrases, paragraphs and texts) providing a set of mutual dependencies. Their identification and proper usage permits the LSA to deal in an excellent way with the synonymy and partially with the polysemy. LSI is a two-stage process that consists of ([DDFLH, 1990], [LFL, 1998], [LSA,1990-1999]): off-line construction of document index, and on-line respond to user queries.

The off-line part of the process is the training part when LSI creates its index. First a large word-to-document matrix X is constructed where the cell (i, j) contains the frequencies of occurrence of the i-th word into the j-th document. After that, a *singular value decomposition* (SVD) is performed which gives as a result three matrices D, T (both orthonormal) and S (diagonal), such that $X = DST^t$. Then all three matrices are truncated in such a way that when we multiply the truncated ones D', S' and T' we obtain a new matrix X' which has the same dimensionalities as X and is the least-squares best fit approximation of X. This results in the compression of the original space in a much smaller one where each document is represented by a vector of low dimensionality (e.g. 100). [BDOKV, 1993].

The on-line part of our search engine (and of LSI) receives the query (pseudo-document) user typed and finds its corresponding vector into the document space constructed by the off-line part using a standard LSI mechanism. Now we can use measure the degree of similarity between the query and the indexed documents by simply calculating the distance between their corresponding vectors.

3 Pre-processing and parsing

This group includes some of the important factors that directly influence all other steps in the process of information retrieval using LSI. The group includes (It is important to say that most of the steps above can be performed in parallel in order to speed up the process): removal of the HTML tags, automatic correction of the mistakes, automatic recognition of the different word-forms (abbreviations, dashed words, different spellings of the same word, word flexions: nouns plural, verb forms etc.), removal of the prepositions and common words, construction of a list of keywords of interest.

Nowadays a considerable part of the information comes from the Web. Thus, the correct recognition and removal of the HTML and scripting elements from the Web pages is an obligatory preliminary step for each information retrieval system.

It is important to note that users entering natural language queries are supposed to be less careful than when using the classic search engines. So, the automatic correction of the common mistakes can lead to significant improvement of the obtained results: usually the query entered is quite short so any typing error is of great importance.

Usually, during the pre-processing step each word is replaced by its main form in the dictionary. A particular problem is caused by the different spellings of the same word caused by the usage of abbreviations or dashed/undashed versions of the same words: e.g. "preprocessing" vs. "pre-processing". Similar problem is caused by a word composed of two or more other ones: e.g. "key word" vs. "keyword" or "key-word". Another common problem that is the different spelling of the same word in UK and US English e.g. "gray" and "grey"). There are two major groups that can be separated:

"–re" vs. "–er",
"–s–" vs. "–z–"

The first group compounds words like "centre" (UK English) and "center" (US English). The second: words like realize vs. "realise" or personalization vs. "personalisation".

Programme packages like Microsoft Office do not care about different word spellings and force the users to choose a relevant variant for the natural language used (e.g. US or UK English). Unfortunately, on the Internet the documents and queries come from all over the world and different spellings may be used. Our experiments on a corpus of religious and sacred texts (see below for details) with a partially automatic mechanism including a pre-prepared list of main forms for some special cases (e.g. irregular verbs) show that the automatic recognition of the different word forms reduces their number by about 30%.

Some authors suggest that morphologically related words belonging to the same root (sharing a common "stem") may be treated alike [CC, 1998]: "investment", "investments", "investor", "investing" and "invest". The stemming can be done either by using the classic algorithm proposed by Porter [Porter, 1980] or by a linguistic dictionary-based morphological analyser. Krovetz [Krovetz, 1993] reports an improvement between 1.3% and 45.3% but recent research by Hull shows that "some form of stemming is almost always beneficial but the average absolute improvement due to stemming is small, ranging from 1 to 3%" ([Hull, 1996]).

The next important pre-processing step includes the removal of some prepositions and common words considered as uninteresting in the parsing process, e.g. "and", "but" and the like for English).

After all above is done we are ready for the next stage: construction of a list of words and whole phrases to be considered as keywords. Sometimes this list is pre-prepared by an expert. More often the word list is automatically constructed during the parsing process. In this case some restrictions are applied limiting: minimum and maximum word length, minimum word frequency in the document corpus, and minimum number of documents required that include the word in question.

Usually the lower bound for word length is set to 3 or 4 and the upper bound to 20 letters. Only words that are common to at least 2 (or possibly more) different documents are considered. That is because a word contained just in a single document cannot contribute to the proximity between the documents and in fact will have almost no effect. This leads to the removal

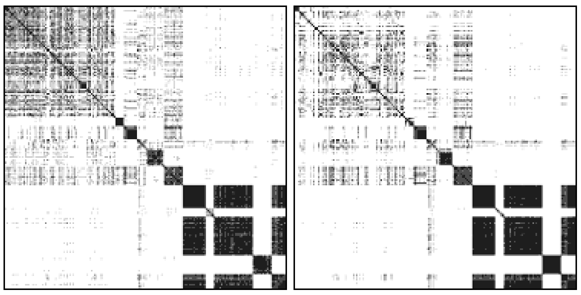
Figure 15.1: Correlation matrix: original and logarithmic

of approximately half of the words, which has an extreme influence on the speed of SVD. On the other hand the application of these techniques prevents us from correctly serving user queries containing the words removed.

For larger corpuses these rules may be even strengthened. In our attempts to create an LSA index on a large collection of Microsoft RFCs (Microsoft technical documentation: 1886 files, 92.8 MB), we found 61451 different words. This number was unacceptable for us because our hardware was unable to perform a singular value decomposition of a matrix sized 1886 x 61541. So, we were forced to remove all one- and two-letter words and to keep only those that are common to at least 5 different documents. As a result the word count climbed to 13858 which allowed us to create the index successfully.

Another possibility is the automatic construction of those lists using a standard statistical procedure. The idea is to identify the most common bigrams and trigrams and to index each of them as whole keyword.

4 Frequency matrix transformations

The next step is the construction of a frequency word-to-document matrix as we have described above. This is the moment to submit the matrix to some transformation. This may include different kinds of normalizations, applying of weights and the like. Common techniques include the usage of logarithm or entropy. Another useful technique is the application of the well known *tf* x *idf* (term frequency times inverted document frequency) weighting system defined as:

$$w_{ik} = \frac{\log(f_{ik}+1)\,\log(N/n_k)}{\sqrt{\sum_{j=1}^{t}\Big[\log(f_{ij}+1)\,\log(N/n_j)\Big]^2}}$$

where f_{ik} is the occurrence frequency of term k in document i, N is the total number of documents and n_k — the number of documents containing term k.

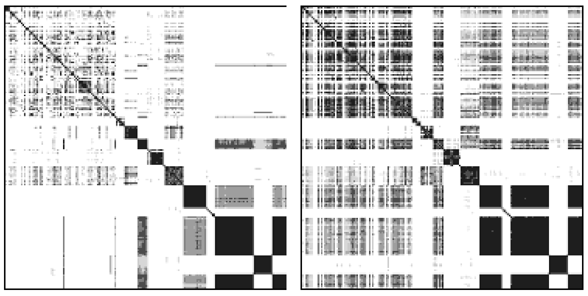

Figure 15.2: Correlation matrix: entropy and entropy&log

Theoretically the best results are achieved when both techniques are applied one after another in the following manner: First, the conversion of the word frequency (+1) for each cell to its logarithm is applied. The values obtained are then used to calculate the entropy for each row (in fact for each key word) using the well-known formula $-p\log(p)$ and the rows are divided by their entropy. The effect of the transformation as a whole is that the most frequent words are weighted lower than the rare ones. So this technique is claimed to emphasize the meaning-bearing words for each passage. ([DDFLH, 1990], [LFL, 1998])

Our experiments show that for large corpuses, like the RFC collection mentioned above, this is beneficial but it is not necessarily the best approach especially for small corpuses when the application of both logarithm and entropy can lead to poor performance. In some cases (see the corpus of religious and sacred texts below) it is better to apply just one of the transformations: the entropy. In other cases the application of logarithm only may give better results. This is obvious from figures 15.1 and 15.2. The right-side matrix on figure 15.2 shows that the application of both logarithm and entropy leads to higher proximity between the apocrypha but introduces a lot of additional noise in the upper right area (see below for details).

Figures 15.1 and 15.2 represent the document correlation matrices (196 x 196) for a corpus of religious and sacred texts (`http://davidwiley.com/religion.html`) obtained using the classic cosine measure. The whole collection contains 1424 different files (21.7 MB) extremely not uniformly distributed by number and size among the different religions. The Old Testament for example contains 928 files (8.91 MB); the New Testament 262 files (4.36 MB) and the Death Sea are just 8 files (22 KB). This disproportion can lead to significant space distortion in the direction of the well-presented big religions. That is why we selected just part of them: 196 religious and sacred texts with 20443 different words. We reduced the words to 11140 keywords by removing the HTML elements and stop-words (using a list of pre-selected 938 stop-words) and keeping only the words common to at least 2 different documents. The religious and sacred texts selected were from 11 categories:

4 kinds of apocrypha (acts, apocalypses, gospels, writings), Buddhism, Confucianism, Dead Sea scripts, The Egyptian Book of Dead, Sun Tzu: The Art of War, Zoroastrianism, The Bible (Old and New Testaments), The Quran and The Book of Mormons. The experiments were made in a 30 dimensional space in 4 different ways by applying or not logarithm and/or entropy. The results are shown in 5 different colors for the five correlation intervals: 87.5–100%, black color; 75–87.5%, dark gray; 62.5–75%, gray; 50–62.5%, light gray; 0–50%, white.

The dark rectangles in the main diagonal show the high correlation between texts belonging to the same religion. For example: the black rectangle from the bottom right corner contains texts from the Book of Mormons. To the left and up on the main diagonal can be found the Quran, then the Old Testament (The Bible), then come the Zoroastrian texts, The New Testament (The Bible), the Sun Tzus Art of War, the Egyptian Book of the Dead and so forth. And the smooth rectangle in the upper left corner shows the relatively high similarity between all kinds of apocrypha present.

What is interesting on figure 15.2 are the two other black rectangles in the last matrix rows showing the high proximity level between the two parts of the Bible and the Book of Mormons. One can consider this surprising. In fact their content is very similar. Let now pass to the left matrix on figure 15.2. We see that by applying the entropy transformation to the base matrix preliminary to SVD we can distinguish between the Old and the New Testament (The correlation is about 70% and the corresponding horizontal rectangle is no longer black but grey. Grey is the rectangle between the Book of Mormons and the Old Testament.) but we cannot distinguish between the New Testament and the Book of Mormons!

5 Choice of dimensionality

The choice of dimensionality to switch to is of particular importance for the further usage of the method and an analysis is needed for each particular case. Although this is a key parameter there are no strict guiding rules for making the correct choice. figure 15.3 contains all 196 singular values found by SVD for our corpus of religious and sacred texts. We can see that the “original” line flattens after 30. If we truncate the dimension further we will lose important factors and if we keep more factors this will result in modeling the unnecessary noise and would guide to poor performance. Different authors suggest as most appropriate dimensions like: 280, 100 or 400. In fact the similarity measure may vary approximately from 50 to 1500.

Figure 15.3 shows also that the choice of dimensionality is dependent upon the matrix transformations performed. Thus, in our example the application of entropy only makes 23 the best dimension choice. If we apply logarithm instead, the best choice is about 12 and when both are applied

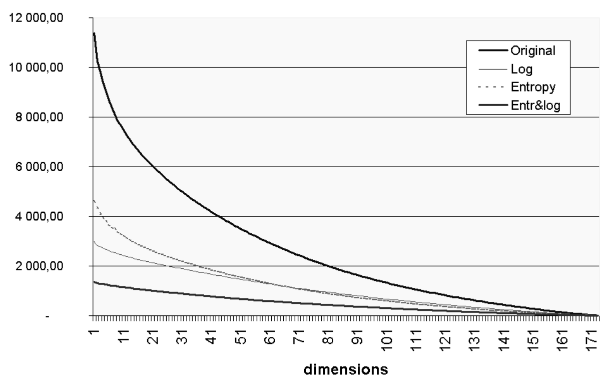

Figure 15.3: Choice of dimensionality

we must switch to 7 dimensional space.

6 Choice of similarity measure

Cosine and angle. As have been mentioned in the beginning of the paper the degree of similarity between two words, two documents or between a document and a pseudo-document (user query) is calculated according to their corresponding vectors. The most frequently used measure of similarity is the cosine between the vector-columns (for documents) or vector-rows (for words).

Thus, the totally dissimilar vectors are orthogonal (cosine is 0) and those corresponding to documents/words with similar meaning are closely aligned (cosine is about 1, but not greater). Higher cosine corresponds to higher degree of similarity. Another variant of this measure can be obtained when using the angle between the vectors instead of their cosine.

Euclidean. Another appropriate measure is the Euclidean distance between the normalised vectors.

The results obtained by the cosine and Euclidean measures are very similar.

Manhattan. Another possibility is the Manhattan distance known also as city-block metric. The application of Manhattan distance is expected to be the worst choice in the general case among all other similarity measures considered in this paper.

Minkowski. The Manhattan and Euclidean metrics belong to the more general class known as Minkowski metrics defined by the formula:

$$d_{ij} = \left(\sum_{k=1}^{r} |x_{ik} - x_{jk}|^s \right)^{\frac{1}{s}}$$

For $s = 1$ we obtain the Manhattan and for $s = 2$ — the Euclidean metric. It is interesting that the results obtained have the property to stabilize for greater values of s. Usually, the sorted document order obtained for $s = 4$ or $s = 5$ is the same as the one for all greater values of s.

Pearson. The product-moment correlation coefficient is one of the most popular similarity measures and our experiments show it to be a good choice for the purpose. The Pearsons coefficient is defined by the formula:

$$d_{ij} = \frac{\sum_{k=1}^{r}(x_{ik} - \overline{x}_i)\,(x_{jk} - \overline{x}_j)}{\sqrt{\sum_{k=1}^{r}(x_{ik} - \overline{x}_i)^2\,(x_{jk} - \overline{x}_j)^2}}$$

Special similarity measures. The classic way to measure the degree of similarity between the query and the indexed documents is simply by calculating the cosine between their corresponding vectors. As a result we obtain a measure of "how central is the discourse to the user query". A search engine may consider useful measures that take in account the length of vectors as well (such as Euclidean distances), which will result in considering "how much has been said about the subject in question" [LFL, 1998]. When indexing Web resources it may be useful to take in account some Internet specific information available. Some authors [VC, 1998] propose the introduction of a number of hyper dimensions (images, anchors, internal links, e-mail or news links, other protocols, java applets etc.) and the application of a similarity measure on both the text and the hyper dimensions. The final similarity measure is defined as their linear combination.

7 Extended Boolean operations

All classic inverted index based search engines give the user the possibility to use Boolean operations to query the indexed documents such as: OR, AND, NOT (AND-NOT), NEAR among with others. This gives the user some kind of flexibility but prevents him or her from using true natural language queries. On the other hand the usage of free form natural language queries is essential for LSI but there is no standard mechanism provided to perform Boolean operations. It is important to note that the classic search engines are able to combine whole phrases with Boolean expressions as well but they are applied to fixed phrases only. So, from LSIs point of view they are applied to just keywords. On the other hand the extended Boolean expressions for LSI we propose combine not just words or phrases but their meaning.

The extended Boolean operations we propose are not limited to search engines and can be used in variety of ways. Consider an e-commerce portal tracking users purchases in order to offer them personalized advertisement: banners, etc. We have a complete history of the customers purchases and want to create a profile for each of them that will allow us to send only relevant information. We can think of the purchases as query components and of the advertisement as new document in the same space. Let us define

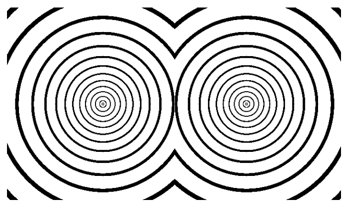

Figure 15.4: OR similarity measure for 2 components

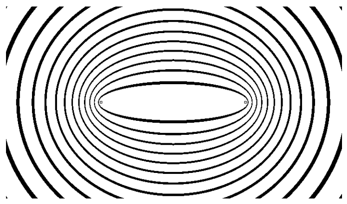

Figure 15.5: AND similarity measure for 2 components

$d_1, d_2, ..., d_n$ as distances (in LSI sense) between the advertisement and the n components of the query. The classic LSI algorithm calculates the cosines between the vectors to find the degree of their similarity. Most of the similarity measures for the Boolean operations we propose below are based on Euclidean distances, although we can use some of the other distances we introduced above. All Boolean operations proposed return a value between 0 and 1. Almost the same results can be obtained using the classic cosines but for some functions it is difficult to fit the values returned in the interval [0,1].

OR-similarity measure. This measure depends only on the minimal distance between the document and the query components and has the following general representation: $S_{or} = f(min(g(d_1), g(d_2), , g(d_n)))$, where $f(x)$ and $g(x)$ are some one-argument functions. In case we have more information for the query we can add weights to the query components and modify $g(x)$ to $g(x, w)$. In case we have additional weights we can write: $S_{or} = f(min(g(d_1, w_1), g(d_2, w_2), ..., g(d_n, w_n)))$

Example: The similarity measure for two-component query, $f(x) = 1/(1 + x)$, $g(x) = x$ is shown on figure 15.4.

AND-similarity measure. This measure depends only on the sum of distances between the document and query components and has the following general representation: $S_{or} = f((g(d_1, w_1) + g(d_2, w_2) + ... + g(d_n, w_n)))$. Usually this measure can be thought as a superposition of distinct similarity measures of the query components.

Combination of the previous two (AND-OR). This similarity measure is a combination between the previous two. $S_{and-or} = f(S_{and}, S_{or})$ We can use a linear combination of S_{or} and S_{and} measures: $S = k.S_{or} + (1 - k)S_{and}$, where k is constant and $0 <= k <= 1$. Figure 15.6 and shows the results for a two- and three component queries with $k = 0.5$.

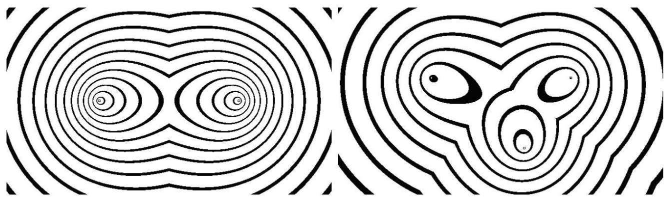

Figure 15.6: Combined similarity for 2 and 3 components, $k = 0.5$

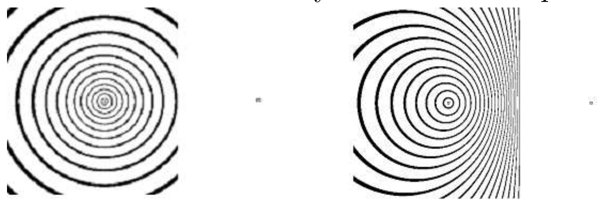

Figure 15.7: MINUS and NOT similarity measures

Binary NOT (AND-NOT)-similarity measure. In case we want to exclude a document we can apply two different similarity measures: MINUS and NOT. For the MINUS similarity measure, if the vector considered is more similar to the exclude vector it will receive a similarity measure of 0 (see the second clause below). Otherwise we return a similarity measure that takes in account the distance to the include vector only. Example: We can use the following MINUS-measure: $S_{not} = d_1$, when $d_1 < d_2$, and $S_{not} = 0$, else. The result is shown on lthe left of figure 15.7.

The problem with this measure is that it takes in account d_2 only when deciding whether to cut the value. A more sophisticated implementation may be used: the NOT (AND-NOT) similarity measure. If the document is more similar to the exclude document text it will receive a similarity measure of 0, but otherwise we return a similarity measure between 0 and 1 that takes in account the distances to both documents. We can use the following NOT-measure: $S_{not} = 1 - d_1/(1 + d_2)$, when $d_1 < d_2$, and $S_{not} = 0$, else. The result is shown on the right of figure 15.7.

8 Conclusion

The usage of LSI is a kind of art and is dependent on a variety of parameters that must be tuned very carefully in order to achieve better results. As was mentioned above for most of them there are no strict guiding rules saying how to choose the appropriate values and in these cases the only option are the experiments. On the other hand when selected carefully they can lead to significant performance improvement. We consider the technique of LSI to be very important in the future and continue our experiments with different kinds of matrix transformation, similarity measures, extended Boolean similarity functions, and their behaviour on different types of texts. Further work concerns study of the mutual dependencies between the different factors and factor-groups as whole.

9 Acknowledgements

Our special thanks go to Jivko Jeliazkov who helped a lot in the experiments and the development of this paper, especially with the Boolean operations.

Bibliography

[AB, 1984] Aldenderfer M., Blashfield R. *Cluster Analysis.* A SAGE University Paper. Sage Publications, 1984

[BDOKV, 1993] Berry M., Do T., O'Brien G., Krishna V., and Varadhan S., *SVD-PACKC (Version 1.0) User's Guide.* April, 1993

[CC, 1998] Caid W., Carleton J. *Visualization of information using graphical representations of context vector based relationships and attributes.* United States Patent 6,794,178. Aug. 11, 1998

[DDFLH, 1990] Deerwester S., Dumais S., Furnas G., Laundauer T., Harshman R. *Indexing by Latent Semantic Analysis. Journal of the American Society for Information Sciences*, 41 (1990), pp. 391-47

[Harman, 1986] Harman, D. *An experimental study of the factors important in document ranking. In Association for Computing Machinery Conference on Research and Development in Information Retrieval.* Association for Computing Machinery, 1986

[Hull, 1996] Hull, D. *Stemming Algorithms: A case study for detailed evauation.* In Journal of the American Society for Information Science. 47 (1):70-84, 1996.

[Krovetz, 1993] Krovetz, R. *Viewing Morphology as an Inference Process.* In Proceedings, 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR93) pages 191-203, Pittsburg, PA, 1993.

[LFL, 1998] Laudauer T., Foltz P., Laham D. *Introduction to Latent Semantic Analysis.* Discourse Processes, 25, pp. 259-284, 1998

[LSA,1990-1999] *LSA* 1990-99, see `http://lsa.colorado.edu`

[Porter, 1980] Porter, M. *An Algorithm for Suffix Stripping.* Program, 14:130-137, 1980.

[Religions] Religions, see `http://davidwiley.com/religion.html`

[SAP, 1990] Simov K., Angelova G., Paskaleva E. *MORPHO-ASSISTANT: The Proper Treatment of Morphological Knowledge.* In: COLING'90, Helsinki, Finland, Vol. 3, pp. 453 - 457, 1990

[The Bible] *The Bible*, see `http://www.bible.org/netbible/download.html`

[The Quran] The Quran, see `http://www.usc.edu/dept/MSA/quran`

[VC, 1998] Vlajic N., Card H. *An adaptive Neural Network Approach to Hypertext Clustering.* University of Manitoba, 1998